Distance Mapping for Corpus-Based Concatenative Synthesis

Diemo Schwarz

DISTANCE MAPPING FOR CORPUS-BASED CONCATENATIVE SYNTHESIS

Diemo Schwarz
UMR STMS
Ircam–CNRS–UPMC
schwarz@ircam.fr

ABSTRACT

In the most common approach to corpus-based concatenative synthesis, the unit selection takes places as a content-based similarity match based on a weighted Euclidean distance between the audio descriptors of the database units, and the synthesis target. While the simplicity of this method explains the relative success of CBCS for interactive descriptor-based granular synthesis—especially when combined with a graphical interface—and audio mosaicing, and still allows to express categorical matches, certain desirable constraints can not be formulated, such as disallowing repetition of units, matching a disjunction of descriptor ranges, or asymmetric distances. We therefore propose a new method of mapping the individual signed descriptor distances by a warping function that can express these criteria, while still being amenable to efficient multi-dimensional search indices like the kD-tree, for which we define the preconditions and cases of applicability.

1. INTRODUCTION

Corpus-based concatenative synthesis matches snippets of sounds in a database of sound segments labeled by audio descriptors, to a target also given by descriptors [1]. We call the segments of sound with their description *units*, and the database of units the *corpus*.

In its many incarnations [1] [2], corpus-based concatenative synthesis is most often based on finding the units closest to the target in a multi-dimensional descriptor space, with the most frequently used approach being a weighted Euclidean distance, where the weight w_i is expressing the relative importance of each descriptor $1 \leq i \leq D$ in the match. This means finding the units closest to the current position x in the descriptor space in a geometric sense, on appropriately scaled dimensions by calculating the weighted square Euclidean distance C^t between x and all $1 \leq j \leq N$ units with

$$C^t(j) = \sum_{i=1}^{D} w_i \, d_i(j) \qquad (1)$$

based on the per-descriptor distance d_i

$$d_i(j) = \frac{\big(x(i) - \mu(j,i)\big)^2}{\sigma(i)^2} \qquad (2)$$

where μ is the (N, D) matrix of unit descriptor data and σ the standard deviation of each descriptor over the corpus. Either the unit with minimal C^t is selected, or one randomly chosen from the set of units with $C^t < r^2$, when a selection radius r is specified, or, third, one from the set of the k closest units to the target.

This paradigm of similarity for unit selection is simple, easy to understand, especially when coupled with a 2D or 3D representation of and interaction with the data, as in the CataRT system [2] for real-time interactive corpus-based concatenative synthesis [3], and can be efficiently implemented using tree-based search indices [4].

The distance-based paradigm can also be applied to categorial descriptors, e.g. a class-based selection like *"start a new sequence of sounds with a unit in the* attack *class, then continue with units from the* sustain *class"* can be expressed by giving integer class-index values as a descriptor, and a sufficiently high weight that pushes any non-matching class far enough away.

The same goes for boolean descriptors that can be used to include or exclude specific units from the selection by giving a binary descriptor and target value and assuring that a non-match will effectively remove the unit from the result set by a high weight.

However, it is difficult to express combined matches of classes (e.g. *"play units from the trumpet or trombone class"*), and certain additional constraints, for instance that all units played in the last p seconds should be different.

Unit Selection by Constraint Resolution Alternative methods based on constraint resolution formulate the unit selection as a CSP (constraint satisfaction problem) [5]. While promising to be more expressive and flexible, the drawback of these methods is that for each new constraint type, code has to be written that integrates it into the constraint solver, and that the CSP itself is NP-complete, such that local search strategies have to be applied to make it computationally tractable, without guarantee to find the best match in a given amount of time. This also means that scalability to larger databases is not assured. Additionally, the CSP approach has only rarely been applied to real-time interactive corpus-based concatenative synthesis [6, 5], and needs a full constraint solver, which is not

[1] A constantly updated overview of the many different approaches, applications, and related work concerning CBCS can be found on http://imtr.ircam.fr/imtr/Corpus-Based_Sound_Synthesis_Survey.

[2] http://imtr.ircam.fr/imtr/CataRT

usually integrated in common real-time interactive sound processing systems.

The aim of this article is thus to reformulate some of the abovementioned additional constraints as a distance-based match, in order to integrate them in commonly used real-time synthesis systems [3, 7]. In section 2 we'll introduce a method of mapping the Euclidean per-descriptor signed distances in order to express unit selection constraints such as avoiding repetition, selecting pitch intervals and chords, and introducing asymmetric distances, detailed in section 3. Section 4 will examine the two cases of how these mapped distances can still be used in conjuction with the efficient kD-tree multi-dimensional search index [4], distinguishing the cases of static and dynamic distance mapping functions.

2. DISTANCE MAPPING FUNCTIONS

Our solution to obtaining more flexibility for expressing various selection criteria while still staying in the distance-based paradigm of unit selection, is to map each individual signed descriptor distance calculation through a distance mapping function $f_i(d, \mathcal{P}) : \mathbb{R} \to \mathbb{R}$ for descriptor i with parameter set $\mathcal{P}$, before calculating the sum in equation (1), replacing d_i from equation (2) by d'_i:

$$d'_i = \frac{\Big(f_i\big(x(i) - \mu(j,i)\big)\Big)^2}{\sigma(i)^2} \tag{3}$$

This allows to pull units in a certain relation to the target closer or push them further away, creating "shortcuts" and, in a sense, folding the descriptor space. Mapped distances can also introduce asymmetry in the distance space, which allows to express lower or upper bounds for selection, as detailed in the following section.

3. APPLICATION EXAMPLES

We will now show several examples of applying distance mapping functions to express unit selection criteria that were not possible using a linear distance alone.

3.1 Range Queries and Note Filtering

For music composition or performance (see for example the applications detailed in [3]), often, the most important criterion is to select precise pitches from a corpus of instrument sounds. Secondary selection criteria could then be, for instance, a certain brilliance, loudness, etc.

With unmapped distances, a given brilliance and loudness target might have been best satisfied with a pitch not matching the given target. Increasing the weight on pitch might also not help here in the general case.

In order to express the priority of the pitch selection, a binary distance mapping function is introduced as

$$f_{\text{range}}(d, r) = \begin{cases} 0 & \text{if } -r \le d \le r \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

This mapping, associated with a high weight, will effectively exclude all pitches outside range r around the target from selection. If the most precise choice within the given range should be favoured, a composite linear and binary distance mapping function can be used:

$$f_{\text{range+}}(d, r) = \begin{cases} d & \text{if } -r \le d \le r \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

Here, we use infinity (in practice, a very high real value) for the non-matching case, in order to still be able to adapt the weights of the within-range match of pitch, or actually any descriptor. See figure 1 for a plot of the range distance mapping functions.

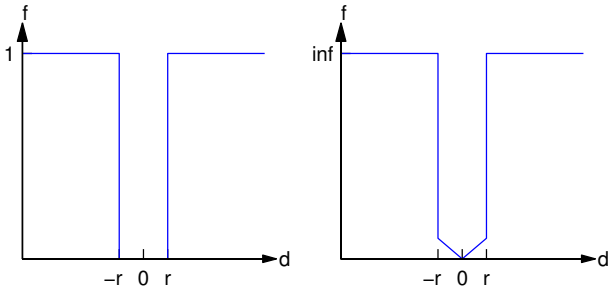

Figure 1. The f_{range} (left) and $f_{\text{range+}}$ (right) distance mapping functions.

This principle can be extended to multiple pitch or category matches by defining a multi-range distance mapping function with notches of size r at distances δ_i as

$$f_{\text{notch}}(d, r, \delta) = \begin{cases} 0 & \text{if } \exists\,\{i \mid \delta_i - r \le d \le \delta_i + r\} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

Again, in order to favour precise matches in the notch centres, we can use

$$f_{\text{notch+}}(d, r, \delta) = \begin{cases} d - \delta & \text{if } \exists\,\{i \mid \delta_i - r \le d \le \delta_i + r\} \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

Figure 2 shows the notched distance functions applied to an octave match, supposing pitch in half-tones. Any chord chould be given here.

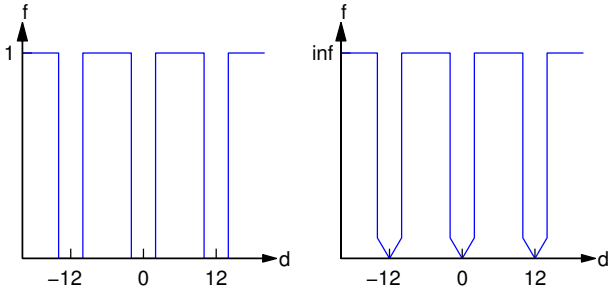

Figure 2. The f_{notch} (left) and $f_{\text{notch+}}$ (right) distance mapping functions.

3.2 Asymmetric Distances

Let's turn now to a case of automated collage or audio mosaicing on a non-uniformly segmented corpus. Here, a given target unit is to be replaced by a unit selected from the corpus. If a continuous output sound is desired, the selected unit should not be shorter than the target unit. It can be longer, however, because a longer database unit can always be cut on playback.

Therefore, we define an asymmetric duration distance mapping function as

$$f_{\text{asym}}(d) = \begin{cases} \infty & \text{if } d < 0 \\ d & \text{otherwise} \end{cases} \tag{8}$$

See figure 1 for a plot of the asymmetric distance mapping function which still prefers a unit matching the exact target duration, to avoid a too large discrepancy between the description of the original unit and the shortened unit.

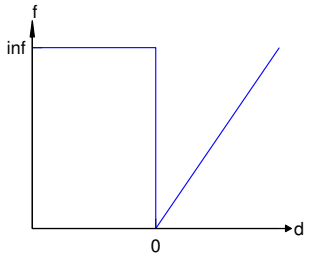

Figure 3. The f_{asym} asymmetric duration distance mapping function.

3.3 Dynamic Taboo Lists

While the above applications were mostly static, and thus could have been implemented by a pre-transformation of descriptor values, we now turn to a dynamic application, where the CSP formulation, or specific programming, [3] used to be the only available solutions: the avoidance of repetitions of played units within a certain temporal window p.

We can formulate this within our distance-based selection paradigm by the introduction of a new descriptor *last played time*, initialised to $-\infty$. When a unit is played, its value is set to the current time. We then keep the target value for this descriptor set to the current time, and introduce an asymmetric distance mapping function for it as

$$f_{\text{taboo}}(d, p) = \begin{cases} 0 & \text{if } d > p \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

[3] As one reviewer correctly remarked, avoiding repetitions could be realised by a time-ordered queue of last played units. However, this wouldn't provide the three following advantages:

First, expressing non-repetition as linear distance allows to express a soft criterion that can be balanced with the distances expressing other selection criteria by weighting, as mentioned for equation (10).

Second, the computational complexity of maintaining and searching the queue for the k units at each selection would add to the kD-tree search, which can already remove the taboo units without additional complexity.

Third, for modular programming environments such as MAX/MSP, an important design criterion is the generality of the modules. Regarding CBCS, this means having a general distance-calculation module that expresses all needs is preferable to a specifically programmed unit selection module.

This allows to calculate in one integrated step the selection criteria on the static database, and the dynamic and history-dependent constraint of avoiding repetitions.

We could now introduce a less strict taboo by allowing to repeat units earlier, when no better choices are available in the corpus, by giving a minimum window $p_{\min}$ and a maximum penalty $d_{\max}$ for a distance mapping function like

$$f_{\text{taboo+}}(d, p, p_{\min}, d_{\max}) = \begin{cases} 0 & \text{if } d > p \\ (d-p)\dfrac{d_{\max}}{p_{\min}-p} & \text{if } d > p_{\min} \\ d_{\max} & \text{otherwise} \end{cases} \tag{10}$$

Figure 4 shows the two taboo distance functions.

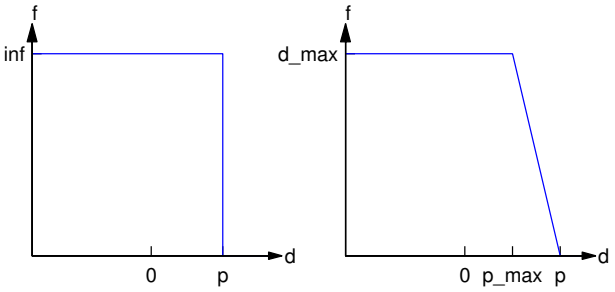

Figure 4. The f_{taboo} (left) and $f_{\text{taboo+}}$ (right) distance mapping functions.

4. INTEGRATION INTO SEARCH INDICES

To perform our distance-based unit selection, and in fact in any application of content-based retrieval, finding the database unit that minimises the target distance C^t is solved most efficiently by a *branch and bound* search algorithm based on the tree-structured index provided by the kD-tree, representing a hierarchical decomposition of the descriptor space by hyperplanes s_n. Details of the indexing and search algorithms are given in previous work [4].

During search, whole subtrees are *pruned*, i.e. discarded from the search, by applying an elimination rule based on the farthest neighbour found so far. This removes a large amount of the distance calculations between feature vectors needed otherwise, resulting in a sublinear time complexity. Several variants of the algorithm are compared by D'haes et al. [8].

We will argue here that our introduction of distance mapping functions does not invalidate the applicability of the kD-tree search index, as soon as certain conditions are met.

First of all, any static distance mapping function that is already applied when building the search index, and whose parameters don't change when searching the tree, is evidently integrated in the index structure.

For the case of dynamically changing distance mapping functions such as in equations (6) or (9), we can define

as necessary condition that the function only *increases* the distance, i.e. iff

$$\underset{\mathcal{P}}{\forall}\ \underset{d}{\forall}\ f(d,\mathcal{P}) \geq d \tag{11}$$

This works, because the elimination rule states that any subtree that lies on the opposite side of its splitting hyperplane s_n when seen from the target point t, and where the distance from t to s_n is greater than the distance d_{kmax} of the farthest neighbour found so far, does not need to be searched. By increasing the mapped distance, we push points further away from the query point, so that the elimination rule still holds. (If we'd decrease the distance and pull points closer, the elimination rule might have already pruned these points from the search, although they might now be part of the nearest neighbours.)

This condition is met for equations (5), (7), and (8), but equations (4) and (6) do not qualify. There is, however, a workaround to make any distance mapping function amenable to the kD-tree search index, which is to start from an all zero distance $d_i = 0$ while building the index, so that any mapped distance will be greater than the original distance. This actually means that descriptor i was excluded from building the search tree, and that the query will use the index on all but descriptor i, and then using the mapped d'_i to sift the result list.

4.1 Influence on the computational complexity

As to the question how the distance mapping influences on the performance of search in the kD-tree, our tests showed only a slight deterioration, with an increase in the number of vector-to-vector comparisons relative to the number of distance mapped descriptors and the parameters of the distance mapping functions.

A detailed evaluation that measures the number of comparisons for a certain number of test cases and situates them between the baseline of brute force search and the optimal kD-tree without distance mapping is planned for future work.

5. IMPLEMENTATION

The distance mapping algorithm described here is implemented as C-libraries and within the mnm.mahalanobis and mnm.knn externals within the free MAX/MSP extension library FTM&CO [9] at http://ftm.ircam.fr, providing real-time optimised data structures, and thus available in the CATART real-time interactive CBCS system [3], and within the MUBU externals [7] at http://imtr.ircam.fr.

In FTM&CO, the distance mapping functions are conveniently and flexibly represented as break-point function (BPF) objects, which also allows to edit them manually in the graphic externals ftm.editor and IMTR Editor [7].

6. CONCLUSIONS

We have seen in this article a simple and efficient way to formulate many additional criteria, that are desirable for musical use of interactive corpus-based concatenative synthesis. These criteria go beyond multi-dimensional proximity and were not possible to be expressed in the habitual framework of unit selection based on Euclidean distances.

Our formulation of distance mapping by a warping function integrates these criteria in the distance calculation while still keeping efficient selection methods based on kD-trees and branch and bound search applicable, with only little loss of efficiency.

The functional formulation of the constraints means that different distance mapping functions could be interpolated to smoothly crossfade from one solution space to another.

Acknowledgments

The work presented here is partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022, http://topophonie.fr.

7. REFERENCES

[1] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, Mar. 2007, special Section: Signal Processing for Sound Synthesis.

[2] ——, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, Mar. 2006, special Issue on Audio Mosaicing.

[3] D. Schwarz, R. Cahen, and S. Britton, "Principles and applications of interactive corpus-based concatenative synthesis," in *Journées d'Informatique Musicale (JIM)*, GMEA, Albi, France, Mar. 2008.

[4] D. Schwarz, N. Schnell, and S. Gulluni, "Scalability in content-based navigation of sound databases," in *Proc. ICMC*, Montreal, QC, Canada, 2009.

[5] J.-J. Aucouturier and F. Pachet, "Jamming With Plunderphonics: Interactive Concatenative Synthesis Of Music," *Journal of New Music Research*, vol. 35, no. 1, Mar. 2006, special Issue on Audio Mosaicing.

[6] J.-J. Aucouturier, F. Pachet, and P. Hanappe, "From sound sampling to song sampling," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, Oct. 2004, pp. 1–8.

[7] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "MuBu & friends – assembling tools for content based real-time interactive audio processing in Max/MSP," in *Proc. ICMC*, Montreal, 2009.

[8] W. D'haes, D. van Dyck, and X. Rodet, "PCA-based branch and bound search algorithms for computing K nearest neighbors," *Pattern Recognition Letters*, vol. 24, no. 9–10, pp. 1437–1451, 2003.

[9] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller, "FTM—Complex Data Structures for Max," in *Proc. ICMC*, Barcelona, 2005.